



SIPN Trust's
HIRASUGAR INSTITUTE OF TECHNOLOGY, NIDASOSHI.

Inculcating Values, Promoting Prosperity
Approved by AICTE, Recognized by Govt. of Karnataka and Permanently Affiliated to VTU Belagavi.
Accredited at 'A' Grade by NAAC
Programmes Accredited by NBA, CMA, ECI, UEL & MBE

Subject: System Software and Compilers (18CS61)

**Module 2: Introduction to Compilers and
Lexical Analysis**

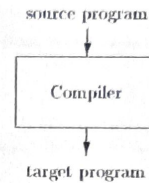
Dr. Mahesh G. Huddar

Dept. of Computer Science and Engineering

Language Processors

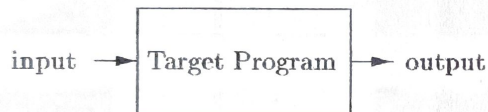
Definition:

- A **compiler** is a program that can read a program in **one language - the source language** - and translate it into an equivalent program in another language - the **target language**



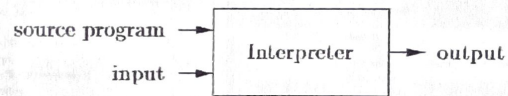
Language Processors

- If the **target program** is an **executable machine-language program**, it can then be called by the user to process inputs and produce outputs



Language Processors

- An **interpreter** is another common kind of **language processor**.
- Instead of producing a **target program** as a translation, an **interpreter** appears to **directly execute** the operations specified in the source program on inputs supplied by the user





Language Processors

- The machine-language (**target program**) produced by a compiler is usually much **faster** than an interpreter at mapping inputs to outputs.
- An interpreter, however, can usually give **better error diagnostics** than a compiler, because it executes the source program statement by statement.

Language Processors

- Java language processors combine compilation and interpretation, as shown in Fig.
- A Java source program may first be compiled into an intermediate form called **bytecodes**.
- The bytecodes are then interpreted by a virtual machine.
- A benefit of this arrangement is that bytecodes compiled on one machine can be interpreted on another machine, perhaps across a network.
- In order to achieve faster processing of inputs to outputs, some Java compilers, called **just-in-time compilers**, translate the bytecodes into machine language immediately before they run the intermediate program to process the input.


Dr. Mahesh Huddar
Course Coordinator


H.O.D
Computer Science & Engg₁
HIT, Nidasoshi



Subject: Object Oriented Concepts (18CS45)

Module 2: Objects and Arrays (C++ Part)

Prof. Mahesh G Huddar

Dept. of Computer Science and Engineering

Arrays and Objects

- Like Structures, It is possible to have arrays of objects.
- The syntax for declaring and using an object array is exactly the same as it is for any other type of array.
- A array variable of type class is called as an array of objects.

Arrays and Objects - Example

```
#include<iostream>
using namespace std;

class Student
{
private:
    int mro;
    char name[20];
    double avgmarks;

public:
    void read_data()
    {
        cout<<"Enter the Roll No.:";
        cin>>mro;
        cout<<"Enter the name: ";
        cin>>name;
        cout<<"Enter the average mark: ";
        cin>>avgmarks;
    }

    void print_data()
    {
        cout<<"Roll No is "<<mro<<endl;
        cout<<"Name is "<<name<<endl;
        cout<<"Average mark"
        <<avgmarks<<endl;
    }
};

int main()
{
    Student st[10];
    cout<<"Enter the student data:"<<endl;
    for (int i=0;i<3;i++)
    {
        st[i].read_data();
    }
    for (int i=0;i<3;i++)
    {
        cout<<endl<<i<<" Student information is:"<<endl;
        st[i].print_data();
    }
    return 0;
}
```

Arrays and Objects - Example

```
#include<iostream>
using namespace std;
class Student
{
private:
    int mro;
    char name[20];
    double avgmarks;

public:
    void read_data();
    void print_data();
};

void Student::read_data()
{
    cout<<"Enter the Roll No.:";
    cin>>mro;
    cout<<"Enter the name: ";
    cin>>name;
    cout<<"Enter the average mark: ";
    cin>>avgmarks;
}

void Student::print_data()
{
    cout<<"Roll No is "<<mro<<endl;
    cout<<"Name is "<<name<<endl;
    cout<<"Average mark"
    <<avgmarks<<endl;
}

int main()
{
    Student st[10];
    cout<<"Enter the student data:"<<endl;
    for (int i=0;i<3;i++)
    {
        st[i].read_data();
    }
    for (int i=0;i<3;i++)
    {
        cout<<endl<<i<<" Student information is:"<<endl;
        st[i].print_data();
    }
    return 0;
}
```

Array inside objects

- An array can be used as a member variable of class.
- Using the instance or object of that class we can initialize the array elements.

Array inside objects - Example

```
#include<iostream>
using namespace std;
class Employee
{
private:
    int empid[10];
    int emp[10];

public:
    void read_data();
    void print_data();
};

void Employee::read_data()
{
    for(int i=0; i<3;i++)
    {
        cout<<"Enter the employee ID: ";
        cin>>emp[i];
        cout<<"Enter the Salary: ";
        cin>>emp[i];
    }
}

void Employee::print_data()
{
    for (int i=0;i<3;i++)
    {
        emp[i]<<endl;
        cout<<"Employee "<<i<<" ID is "<<
        emp[i]<<endl;
        cout<<"Employee "<<i<<" Salary is "<<
        emp[i]<<endl;
    }
}

int main()
{
    Employee emp;
    emp.read_data();
    emp.print_data();
    return 0;
}
```

Mahesh G Huddar
 Dr. Mahesh Huddar
 Course coordinator

Mahesh G Huddar
 H.O.D
 Computer Science & Enyy.
 HIT, Nidasoshi



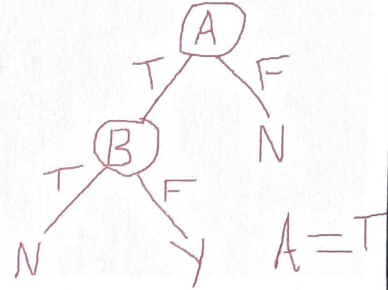
Decision Trees

Prof. Mahesh G Huddar

Dept. of Computer Science and Engineering

3.1 Give decision trees to represent the following boolean functions:

- (a) $A \wedge \neg B$
- (b) $A \vee [B \wedge C]$
- (c) $A \text{ XOR } B$
- (d) $[A \wedge B] \vee [C \wedge D]$



Decision Tree for Boolean Functions

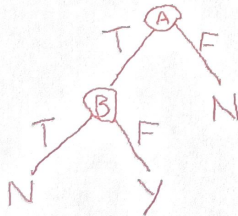
- (a) $A \wedge \neg B$
- (b) $A \vee [B \wedge C]$
- (c) $A \text{ XOR } B$
- (d) $[A \wedge B] \vee [C \wedge D]$

Decision Tree for Boolean Functions

- Every Variable in Boolean function such as A, B, C etc. has two possibilities that is True and False
- Every Boolean function is either True or False
- If the Boolean function is true we write YES (Y)
- If the Boolean function is False we write NO (N)

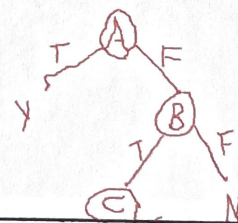
Decision Tree for Boolean Functions

- (a) $A \wedge \neg B$



Decision Tree for Boolean Functions

- (b) $A \vee [B \wedge C]$



MH 2

Dr. Mahesh Huddar

Course coordinator

MH

H.O.D

Computer Science & Engg.
 HIT, Nidasoshi